Should Your COI Use Cursor-On-Target?

Dr. Scott Renner
The MITRE Corporation
sar@mitre.org

I'm going to assume that you're asking this question, which means you already know about the Net-Centric Data Strategy, and about COIs, and what Cursor-On-Target is. Now, I can't answer the question for you, because I don't know what your COI is all about. But I can tell you some things that will help you make a good decision.

1. Every COI should consider the CoT "loose-coupler" approach.

This approach calls for establishing a small shared vocabulary that covers only the data to be exchanged in the present spiral, thereby minimizing the total amount of vocabulary learning required. The participants focus on the intersection of their information needs, not the union of those needs.[1] This "loose-coupler" shared vocabulary does *not* have to be adopted internally by the participating applications; it is instead used as an intermediate exchange format, or as a reference schema for data mediation.

Most new COIs are expected to produce useful results in a very short time, typically nine to twelve months. They do not have time to develop a large comprehensive vocabulary. They may not even have time to learn an existing vocabulary if it is too large and cannot easily be broken into subsets. Therefore, your COI should seriously consider following the CoT *approach*, even if you eventually decide not to use the CoT *schema*.

2. The CoT "what/where/when" concept is widely applicable

The CoT developers looked at a number of battlefield data exchanges that are described in the standard message formats (TADIL, VMF, USMTF, etc.). They found that a large majority of these exchanges are reports of an event that occurs (or thing that exists) at a particular place and time. This "what, where, and when" information is the core of the CoT schema. It turns out that many important data exchanges can be partially described in terms of "what, where, and when."

This same core data can also be used as discovery and distribution metadata. That is, even if your applications are not directly interested in what, where, or when, they may still be able to use those three things to describe the data assets they *do* need, perhaps in a search query, perhaps in a subscription that pushes those data assets to them.

Of course, there are data exchanges that are not a good fit to the CoT what/where/when concept. For example, MITRE spent some time trying to represent the data exchanged between GMTI producers within the CoT model. The results were unsatisfactory. CoT

[1] Here the term "intersection" is not used with mathematical rigor. It's more of an analogy. Determining the right boundaries of a vocabulary is an art, not a science.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
30 JAN 2013	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Should You Use Cursor on Target (CoT)	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Cursor-on-Target, AFLCMC/HNBC Architecture & Standards Division**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**10**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

is intended to describe a single what/where/when event; GMTI producers want to describe a radar "dwell" that contains many separate what/where/when target reports. It's not impossible to do this within CoT – technically, nothing is – but the fit was so unnatural that the effort was dropped.

There is a very good chance that the CoT what/where/when concept is relevant to part or all of your COI's data needs. This can be true even if you eventually decide not to use the CoT schema. It's definitely worth a good, hard look.

3. The CoT "when" data model is widely applicable

The data model used by CoT for "when" data has two parts: the data type used to capture a time, and the specific times that are captured. For the data type, CoT chose the *datetime* type from XML Schema. CoT captures the time of the report as a point in time, and the time of the event as an interval of time (with starting and ending points). All of these choices are applicable to many subject-area domains:

- The XML Schema *datetime* type is widely understood by XML parsers, is derived from ISO 8601, and can represent points in time of arbitrary precision. By adding a profile forbidding the use of "local times" (without timezones), all times become UTC. (All known CoT implementations conform to such a profile.)

- It is often useful to capture the time at which a message was created. Representing this time as a single point makes sense, because the generating application typically knows exactly when this occurs, and the length of time needed to generate the message is usually unimportant.

- On the other hand, the time of a reported event is often represented as an interval of time, because the exact time may not be known, and the duration of the event is often important. (It may be important to know which of these is being represented; see #9.)

If your COI needs to specify the exact time of a report, or the time interval during which an event occurs, then the CoT "when" data model is very probably a good fit. Unless you are operating in a severely bandwidth-constrained environment, the CoT data type will also fit well. You may use it both for data you produce ("the FOO was running at these times") and for data requests ("tell me which FOOs were running at these times").

4. The CoT "where" data model is widely applicable

The CoT schema uses five data elements to capture "where" data. The first three describe a point in three-dimensional space; the second two describe the radius and half-height of a cylinder centered at that point. Together, these data elements describe a precisely-located cylinder whose axis runs through the center of the earth. This data model can be used to describe:

- 3-dimensional locations with given error estimate or known precision
- 2-dimensional location of ground objects
- 2-dimensional location of air objects with unknown altitude
- precise location of objects or events that are shaped like a cylinder
- discovery/distribution metadata for any sort of object or event (i.e. "if you're interested in FOO objects related to this location, then you'll be interested in this object.")

In short, you can use this data model to represent many kinds of "where" data. (Of course, it is sometimes critically important to understand exactly *which* kind of "where" data is being represented. The CoT schema supports any of them, and relies on context to sort them out.)

CoT chose the WGS84 coordinate system because it is native to the GPS system. Latitude/longitude (in decimal degrees) and height-above-ellipsoid (HAE) were likewise selected because this representation is supplied at the interface of almost every GPS receiver. The vast majority of geospatial web services also specify input and output locations in terms of lat/long.

If your COI needs to specify the geolocation of an object or event, then the CoT "where" data model is likely to be a good choice. You may use it both for the data you produce ("here is the location of the FOO") and for data requests ("tell me about all the FOOs at this location").

Of course, there are some situations where the CoT data model and/or data types don't work well. Here are three examples:

- Some applications compute a new location from an input location. ("If it's located at X, and it's moving with vector V, then later on it will be at Y.") Such computations are rarely performed in a lat/long/hae coordinate space. Instead, the location is first converted to an x,y,z coordinate space, where the computations are much simpler. If your COI needs to perform a series of these computations, distributed over several participants, then you won't want to use the CoT data types for your data exchange, because you don't want to do the coordinate conversion over and over – it's slow, and it loses precision. The CoT data model is still ok, but you'll want to replace the data types with the WGS84 ECEF coordinate system. JAMD is an example of this sort of COI.

- Some applications need to use a different geometric shape to describe locations. For example, with GMTI data an ellipse is more suitable than a (zero-height) cylinder. In other applications, pressure altitude is a better representation of height than HAE. If your COI is like these, then you may not want to use the CoT data model for geolocation. (You might still want to use the CoT model for discovery/distribution metadata, and then also supply the "real" location using a different model.)

- Some applications have a completely different view of location. For example, the CoT model is not a good replacement for mailing address, because it doesn't represent data

that is important for carrier routing and sorting. CoT isn't a good model for relative locations ("in the supply cabinet, 3rd shelf"). Of course, these aren't *geolocations*, so you really shouldn't expect CoT to work for these kinds of "where"…but if I didn't bring it up, somebody else probably would.

The CoT documentation in the *Developer's Guide for CoT* [1] provides semantics for the "where" cylinder that are precise but perhaps not what your COI needs. The size of the cylinder is defined to completely enclose the true location of the event with a one-sigma certainty. That is, the event is actually located within the cylinder about 67% of the time.[2] For point locations, if your applications require a different certainty, they can easily compute the dimensions of the corresponding cylinder.[3] If this is not what your COI needs, you can always choose different semantics for the cylinder in your COI – but if you do, you won't be as interoperable with other, CoT-based COIs.

I once wondered whether a cone would be better than a cylinder for the "where" shape, because with a large (e.g. 100-kilometer) radius you get some strange effects related to the curvature of the earth. However, reading between the lines in the *Developers' Guide*, I concluded that the cylinders are expected to be small, commensurate with the sort of error bounds one would expect to obtain with a GPS receiver.[4] With small cylinders, curvature may be ignored and your computations become simple and cheap. However, if your COI actually needs true cylinders that span large areas, then you may need to worry about the curvature effects, and may not be interoperable with other CoT-based COIs.

5. The CoT method of a simple core model with optional extensions is useful

The CoT schema is designed as a package with a simple core schema and several optional extension schemas. CoT messages must contain all of the core fields; they may contain zero or more extension data sets in the "details" field. The core schema provides the most basic what/where/when data. The extensions provide additional detail that may be used by some applications and ignored by others (when not needed, or not understood).

This pattern can be applied quite widely. In general, we should expect to see a few simple vocabularies that are used by large groups, and a greater number of detailed vocabularies that are used by small groups. We would like to see these arranged in a hierarchy, so that the detailed vocabularies are extensions of the simple vocabularies, and

[2] Or possibly 39% of the time. The CoT documentation doesn't say whether they are assuming a linear or two-dimensional error model for the cylinder radius. This matters, because the probability rating of "one sigma" is different when two parameters (latitude and longitude) are involved [2]. If 39% vs. 67% makes a difference in your COI, either supply more semantics in your COI vocabulary, or make sure each consumer knows the interpretation each producer is using.

[3] For events that are volumes instead of points, you must extend the CoT core if you want to compute different certainty volumes. This hasn't been a problem in any real-world CoT application. See the CoT Developers' Guide for more information.

[4] Inferred from a description of the algorithm used to determine whether two cylinders intersect: fast, efficient, practical for the CoT applications I know about… but wrong for large cylinders.

the small groups are subsets of the large groups. The CoT schema design is a rough match to this pattern.

If your COI fits this pattern, then you may also want to design your data schemas around a small core and a number of detailed extensions. This method is especially useful if your COI will use some sort of publish-subscribe mechanism to exchange data.

6. The CoT schema is a good engineering compromise for a large class of applications

The CoT designers did not set out to satisfy the whole universe of data exchange. They made design choices that suit a particular class of applications. The key discriminators are:

- Flexibility is especially important: The CoT schema can be quickly repurposed or extended to support a new data exchange need. This flexibility depends on people who understand and can supply the context of each data exchange.

- Bandwidth is important, but not crucial: The CoT schema reduces the size of the runtime data by using XML attributes instead of elements, by choosing short tag names, and by squeezing all the parts of its "what" taxonomy into a single string value. When bandwidth is not a concern, these choices are unnecessary. When bandwidth is truly crucial, XML is squeezed out in favor of hand-crafted binary formats.[5]

- Solutions are required for immediate use: The CoT designers wanted to implement and deploy their work right away. Future technology (XQuery, EXI, etc.) need not apply for consideration.

If your applications need what/where/when data, and they fit the description in this section, then the CoT schema is probably a good match for your COI. If the above discriminators do not apply to your applications, then you may have trouble with the CoT schema. The rest of the paper describes some things that may bite you.

7. The CoT core schema cannot accommodate arbitrary "electronic tearline" security markings

Security markings indicate the classification level of the marked data. These are important in determining who may be allowed access to the data. An "electronic tearline" is one way to produce multiple versions of the same information at varying classification levels. The name comes from the practice of preparing a paper document so that the classification level strictly decreases from top to bottom, and then tearing off the top portion which a consumer is not cleared to receive and giving him the rest. With

[5] This is true at present, but perhaps not for much longer. In November 2005 the W3C started the *Efficient XML Interchange (EXI)* working group to define an alternative binary encoding of the XML Information Set. There is at least one prototype implementation that produces encodings equivalent to the best hand-crafted formats.

paper documents, the author must "write for release", in effect preparing multiple versions so that each section of the document makes sense without the more-classified sections. With XML data, the developers must design for release, to ensure that data at different classification levels can be separated and labeled in a standard fashion [3]. This is mandatory practice for terrorist-related information [4] and is good practice in any multiple-security-level environment.

CoT can be used in such an environment, but only if the "tear-off" information is completely contained in the location and/or detail element. It is not possible to assign different security labels to the information in the CoT core. For example, there is no way to indicate that the time of an event is unclassified, while saying that the identifier of the event is secret.[6] This is a consequence of the decision to represent the core information using XML attributes instead of XML elements.

If your COI needs the core to be part of its "design for release" in a multiple-security-level environment, then you will not want to use the CoT schema. You may want to use the *Common Battlespace Object* instead [5]. The CBO schema captures all of the information in the CoT schema while conforming to the separation and labeling standards for terrorist-related information. (However, it uses many more bits to capture the same information.)

8. The CoT taxonomy may not be a good fit

The CoT designers have created a taxonomy of the events and things that can be described by a CoT message. In essence, this is their taxonomy of "what". At the top level, this taxonomy specifies five categories:

- *atoms* – describes a physical thing
- *bits* – describes a digital data asset
- *reservation* – describes an area of space (occupied, contaminated, etc.)
- *tasking* – describes a request for service
- *capability* – describes the ability to do something within an area (e.g. rescue)

The *atoms* category is further divided into subcategories that are derived in part from MIL-STD-2525B, *DoD Common Warfighting Symbology*. This means that CoT messages in the *atoms* category can be interpreted as "draw this symbol at this location on your map display", which can be handy. (Of course, the message may be intended to mean something different, or more.) The origin of terms in the rest of the CoT taxonomy is not described. I believe it is *ad hoc.*

Every taxonomy is a hierarchy. Here we find a universal problem: Hierarchies are great, if it's *your* hierarchy. Otherwise they stink. There is no "universal" taxonomy that suits everybody [6]. People really don't like shoehorning their information into some alien

[6] Earlier drafts of this paper said you couldn't have an unclassified location and a secret identifier. Wrong. CoT handles that OK, because there's a separate "point" element for location.

conceptualization of the world. If you try to make them do it anyway, why bother having COIs?

It may be that the CoT taxonomy is a good fit to the way your COI looks at the world. Then again, it may not be. For example, are individual people important in your COI? Do you naturally see those people as a subcategory of "atoms"? The first, most important thing you want to know about a person: is that whether he is friendly, neutral, or hostile? The second: whether he is operating in space, in the air, on the ground, in or under the sea? If these sound like strange questions, if this is not how you categorize people, then the CoT taxonomy may not be right for you.

If your COI can adopt the CoT taxonomy, this will improve (but not guarantee) interoperability with CoT-enabled systems. If the CoT taxonomy doesn't fit your COI, you may create your own, while still using the CoT schema. (Here again you may want to use the CBO schema, which has explicit support for multiple "where" taxonomies.)

9. The CoT schema may be *too* flexible for your COI

On first glance this criticism seems unfair. The CoT schema is very *inflexible* in certain datatype choices. If you want to play, you must use WGS84 lat/long/hae for location. This makes life much easier for the consumers.

However, the CoT schema is very flexible in terms of the data exchanges it can support. That is, it may be quickly repurposed to support new and modified data exchanges. This flexibility is possible because the schema does not capture completely detailed semantics. These are left to people who understand the data exchange context.

For example, consider the data model for "when" information. Do you want the "start" and "stale" elements to denote a precisely-known interval of time? Do you want them to denote the error bounds of an imprecisely-known instant? You may use the CoT schema for either of these. But here's the thing: when the consumer gets your data, he may need to know *which one of these* you chose. He can't tell from the data, or from the schema definition. He has to figure it out from the data exchange context.

Other semantic aspects are equally flexible. The "when" information – does that represent actual takeoff time? The time the cabin door is shut? Scheduled boarding time? It can be any of these. You can't tell from the data, or from the CoT schema documentation. You must understand the exchange context.

CoT is so good at this kind of flexibility that I sometimes think of it as a data exchange *toolkit*, or perhaps a *framework* for making quick data connections – like a plumber's bag of parts and tools for fitting pipes together in the field, when the plans aren't right and the existing pipes don't go where you need them and you don't have time to send the whole thing back to the factory. Of course the plumbers understand the context of what they're rigging up, and of course they prefer flexible parts they can fit into the context at hand. You're always going to need to do this sort of thing in net-centric operations.

However, this sort of flexibility doesn't only solve problems; it can create them, too. You often run into trouble when you try to keep a jury-rig running for a long time, after the plumbers who built it have gone home. Or when you try to make them so big that the plumbers lose track of what they're trying to do. Or when you need more than one rig, each plumber thinks he's the only guy on the site, and they start getting in each other's way.

There are COIs in the commercial world that do capture detailed semantics in their schemas. These COIs need scale and stability more than flexibility. For example, the *Health Level Seven (HL7)* consortium invests a lot of effort to ensure that everyone can tell how to handle an "admission, discharge, and transfer" (ADT) message from the schema documentation alone, without knowing anything else about the context. Then they put in some more work on the foundation for conformance claims – if Fred says his system handles the message correctly, and Bill says it doesn't, they've captured enough semantics so that a neutral third party can judge who's right. All this supports an environment with hundreds of developers and thousands of participants, in which the information exchanged is known in advance, but who you exchange with is not. So when your application gets an ADT message from anywhere, it knows exactly what it means, and what to do. The context is captured in the schema definition.

The CoT schema doesn't support that level of context-independence. If your application gets a CoT message, and all you have is the semantics captured in the CoT schema, then you don't really know what to do with it. If that's a problem in your COI, then you'll need something less flexible, something more thoroughly defined than CoT.

10. Don't count on seamless interoperability with other CoT-based models

This is a corollary of #9. If my COI vocabulary is CoT-based, and your COI vocabulary is CoT-based, can your application consume messages from my applications? Maybe, maybe not. You'll probably be able to parse them, and display something on a map, and otherwise present some data to an operator. But will your application be able to take the data and do the right thing with it? You can't be sure.

Conclusion

Cursor-on-Target produced a surprising amount of controversy over the past few years. I've found people who think that every system should use it. I've found people who want it stamped out. (Amusingly enough, both sides placed me with the enemy at one time or another.)

I think the controversy comes from misunderstanding the scope of CoT. Pretty much everybody has some potential use for CoT. So it's universally applicable. Pretty much everybody has some data tasks for which CoT is not suited. So it isn't universally applicable. It is, it isn't, it is, it isn't… easy to get that argument going, unless everybody listens hard and is very careful to say precisely what they mean.

In this paper I've tried to dissect CoT and show the utility and the limitations of its different aspects.

- You should certainly consider the CoT "loose-coupler" approach
- The "where" and "when" data models are worthy of reuse in many situations
- Something like the CoT hierarchy of a core and extension schemas is useful
- The specific CoT taxonomy may not be what your COI needs
- You may need less flexibility and more semantics in your COI vocabulary

"Should you use CoT?" is not really a yes-or-no question. Your COI may use part or all of CoT, as it fits your needs. The discussion in this paper was intended to help you judge how well the various parts fit.

Acknowledgements

Arnie Rosenthal provided helpful comments. Mike Butler helped to correct a few errors in the first draft. Of course, any remaining errors are my fault, not theirs.

References

[1] Mike Butler, *The Developer's Guide to Cursor on Target,* August 2005. http://cot.mitre.org/pub/CoT/Development/Developers_Guide_to_CoT_V4.pdf

[2] U.S. Army Corps of Engineers, *Engineering and Design – NAVSTAR Global Position System Surveying*, EM 1110-1-1003, July 2003, chapter 4, pp. 5-6. http://www.usace.army.mil/publications/eng-manuals/em1110-1-1003/c-4.pdf

[3] Office of the Director of National Intelligence, *Common Terrorism Information Sharing Standards (CTISS) for Tearline Applications: XML Implementation*, Release 1.0, November 2004. https://www.icmwg.org/ciss/introduction.asp

[4] Executive Order 13388, *Further Strengthening the Sharing of Terrorism Information To Protect Americans,* October 2005. http:// edocket.access.gpo.gov/2005/pdf/05-21571.pdf

[5] FIOP SEWG Data Engineering Team, *The Common Battlespace Object (CBO), Executive White Paper,* September 2005 (draft).

[6] Cory Doctorow, *Metacrap: Putting the Torch To Seven Straw-Men of the Meta-Utopia,* August 2001. http://www.well.com/~doctorow/metacrap.htm

Acronyms

CBO	Common Battlespace Object
COI	Community of Interest
CoT	Cursor on Target

ECEF	Earth Centered Earth Fixed
EXI	Efficient XML Interchange
FIOP	Family of Interoperable Pictures
GMTI	Ground Moving Target Indicator
GPS	Global Positioning Satellite
HAE	Height Above Ellipsoid
HL7	Health Level 7
JAMD	Joint Air and Missile Defense
SEWG	System Engineering Working Group
TADIL	Tactical Digital Information Links
USMTF	United States Message Text Format
UTC	Coordinated Universal Time
VMF	Variable Message Format
W3C	World Wide Web Consortium
WGS84	World Geodetic System 84
XML	Extensible Markup Language